**Student Name:** **Student ID:**

# Objectives

* Outline the basics of the Immunity debugger.
* Find exploit-friendly instructions.
* Describe bad character filtering.
* Defeat anti-debugging code in malware.

# Important Information

* For *every* lab and home assignment, store all your work in your personal repository in a subdirectory named **mXX**, where XX is the module number. Carefully name the program as described in each problem.
* Your programs are extracted from your repository by a Python script. If there are any errors in the program name, then your instructor will never see your program, and you will receive a mark of zero.
* Push your work to the server often, and ensure that you push the final version of a program by the deadline specified, because the script extracting them can be run at any time after the deadline.

# Instructions

1. Read Chapter 5 in the *Gray Hat Python* textbook. The following links are also useful:
   * <https://docs.python.org/3/library/pdb.html>
   * <http://www.gnu.org/software/gdb/documentation/>
   * <http://sourceware.org/gdb/current/onlinedocs/gdb.pdf.gz>
   * <https://docs.python.org/2/extending/extending.html>
   * <https://support.microsoft.com/en-us/kb/875352>
   * <https://www.sans.org/reading-room/whitepapers/malicious/basic-reverse-engineering-immunity-debugger-36982>
   * <https://sgros-students.blogspot.ca/2014/05/immunity-debugger-basics-part-1.html>
2. Complete the problems below.

**Note:** Problem numbering continues from the module’s lab.

## Problem 10

Go back to Problem 5 in the lab. This time, rather than hard-coding the WinExec() call, locate its address at runtime.

## Problem 11

1. Write a C program named **overflow.c** that generates a buffer overflow.
   1. To accomplish this, declare a local variable in your main() of at least 512 bytes.
   2. Have your **main()** call a function **fun()** and then declare a local stack variable in that function.
   3. Use shellcode that executes calc.exe.
2. Find a suitable address for a “jmp esp” instruction and insert it before the shellcode.
3. Apply the shellcode and copy it into the buffer.
4. Execute the program and verify that the calc.exe is spawned.

## Problem 12

Review the **badchar.py** code in Chapter 5 of the *Gray Hat Python* textbook and fix it so that it performs as intended.

## Problem 13

1. Bypass DEP on Windows by reusing the code in overflow.c and modifying it for this purpose.
2. Use the Immunity Debugger script **findantidep.py**.
3. Verify that the DEP has been bypassed.